(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
22 juillet 2004 (22.07.2004)

PCT

(10) Numéro de publication internationale
WO 2004/061622 A2

(51) Classification internationale des brevets[7] : G06F 1/00

(21) Numéro de la demande internationale :
PCT/FR2003/003805

(22) Date de dépôt international :
18 décembre 2003 (18.12.2003)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
02/16932 24 décembre 2002 (24.12.2002) FR

(71) Déposant *(pour tous les États désignés sauf US)* : **TRUSTED LOGIC** [FR/FR]; 5, rue du Bailliage, F-78000 Versailles (FR).

(72) Inventeurs; et
(75) Inventeurs/Déposants *(pour US seulement)* : **HAMEAU, Patrice** [FR/FR]; 18, rue de Belle Feuille, F-92100 Boulogne Billancourt (FR). **LE METAYER, Daniel** [FR/FR]; 23, rue de la Celle, F-78150 Le Chesnay (FR).

(74) Mandataire : **DE SAINT PALAIS, Arnaud**; Cabinet Moutard, 35, rue de la Paroisse, F-78000 Versailles (FR).

(81) États désignés *(national)* : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) États désignés *(régional)* : brevet ARIPO (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Déclaration en vertu de la règle 4.17 :
— *relative à la qualité d'inventeur (règle 4.17.iv)) pour US seulement*

Publiée :
— *sans rapport de recherche internationale, sera republiée dès réception de ce rapport*

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

(54) **Title:** METHOD OF SECURING COMPUTER SYSTEMS COMPRISING A CODE INTERPRETATION MODULE

(54) **Titre :** PROCEDE POUR LA SECURISATION DES SYSTEMES INFORMATIQUESINCORPORANT UN MODULE D'INTERPRETATION DE CODE.

(57) **Abstract:** The invention relates to a method of securing computer systems comprising at least one code interpretation module and memory capacity for storing the code to be interpreted. For said purpose, the invention consists in making more difficult attacks involving physical measures and/or requiring a synchronisation with the interpreted code, by introducing variants into the interpreted code runtimes and the measurable physical prints.

(57) **Abrégé :** Le procédé selon l'invention concerne la sécurisation des systèmes informatiques comportant au moins un module d'interprétation de code et des capacités mémoire de stockage du code à interpréter. Elle propose à cet effet de rendre plus difficiles les attaques reposant sur des mesures physiques et/ou nécessitant une synchronisation avec le code interprété en introduisant des variantes dans les temps d'exécution des codes interprétés et les empreintes physiques mesurables.

WO 2004/061622 A2

PROCEDE POUR LA SECURISATION DES SYSTEMES INFORMATIQUES INCORPORANT UN MODULE D'INTERPRETATION DE CODE.

La présente invention concerne la sécurisation des systèmes informatiques comportant au moins un module d'interprétation de code et des capacités mémoire de stockage du code à interpréter.

Elle a plus particulièrement pour objet de résoudre les problèmes de sécurisation des systèmes informatiques comportant au moins un module d'interprétation de code (un code étant défini comme un ensemble structuré d'instructions) qu'on appellera simplement "interpréteur" par la suite (interpréteur matériel : micro-contrôleur, micro-processeur ou logiciel : machine virtuelle) et des capacités mémoire de stockage du code à interpréter (ou "code interprété").

Ledit code peut être écrit directement par un programmeur, être obtenu de manière automatique (ce qu'on appellera la "génération de code") à partir d'un "code source" dans un langage qui est généralement de plus haut niveau ou encore résulter d'une combinaison de production automatique et d'interventions manuelles.

D'une façon générale, on sait que la plupart des attaques recensées contre de tels systèmes informatiques reposent sur des mesures physiques (émission électromagnétique, etc.) lors de l'exécution et nécessitent la synchronisation avec le code interprété. En d'autres termes, il est nécessaire à l'intrus de

déterminer à quel moment l'interpréteur se trouve en train d'exécuter certaines fonctionnalités du code. Parmi ces techniques les plus connues, on peut citer celles développées pour retrouver une clé dans des algorithmes cryptographiques par espionnage passif de l'émission physique d'un circuit : les attaques de type SPA ("Simple Power Analysis") et DPA ("Differential Power Analysis") en particulier ont été utilisées avec succès pour découvrir des clés DES ("Data Encryption Standard"). A titre d'exemple, sur une plate-forme Java embarquée ("Java Card", "JEFF", "J2ME", ...), ces attaques peuvent être utilisées afin d'essayer d'obtenir des informations sur des secrets manipulés par la machine virtuelle Java. Ces secrets peuvent concerner aussi bien les données confidentielles que le code Java lui-même.

L'invention a donc plus particulièrement pour but de supprimer ces inconvénients.

Elle propose, à cet effet, de rendre plus difficiles les attaques reposant sur des mesures physiques et/ou nécessitant une synchronisation avec le code interprété en introduisant des variantes dans les temps d'exécution des codes interprétés et les empreintes physiques (par exemple, et de manière non exclusive, émission électromagnétique, etc.) mesurables.

Selon l'invention, ce procédé fait intervenir essentiellement deux types de variantes dans les temps d'exécution des codes interprétés, de la manière suivante :

- en provoquant à certains endroits d'un code interprété des dérivations vers de nouvelles portions de code (qui n'appartiennent pas au code d'origine) destinées à compliquer la synchronisation et l'empreinte physique de l'exécution, ou
- en proposant une pluralité de mises en oeuvre de certaines instructions, chacune réclamant un temps d'exécution différent et/ou présentant une

empreinte physique différente et fournissant un résultat identique, en faisant en sorte que deux exécutions de cette instruction au sein d'un même code puissent être réalisées par deux mises en oeuvre différentes.

Ainsi, en introduisant des distorsions dans les temps d'exécution et en modifiant l'effet physique de l'exécution, les deux types de variantes ci-dessus rendront plus difficile toute tentative de corrélation entre les manifestations physiques observées d'un code interprété et ses fonctionnalités.

Avantageusement, ce procédé permettra de rendre le code apparemment exécuté différent à chaque exécution, et rendra donc plus difficile la découverte du code réel de l'application.

Ce procédé pourra faire intervenir :

- pour la première variante :
  - deux modes d'introduction de "codes de dérivation",
  - quatre modes de réalisation de "codes de dérivation",
- pour la deuxième variante :
  - deux modes d'introduction de "mises en œuvre multiples" de certaines instructions,
  - trois modes de réalisation de "codes alternatifs" à empreinte physique et durée variable.

Concernant la première variante, le premier mode d'introduction de "codes de dérivation" consiste à introduire une (ou des) instruction(s) spécifique(s), dite(s) "de dérivation" à certains endroits particuliers du code. Cette introduction peut être réalisée soit manuellement, soit automatiquement lors de la génération de code. Dans le dernier cas, le générateur de code peut être guidé, pour produire ces instructions, par des annotations insérées par le programmeur dans le code source et permettant de désigner des portions de code sensibles (par exemple, et de manière non limitative, des procédures de

chiffrement ou de vérification de droits d'accès). L'exécution d'une instruction de dérivation par l'interpréteur provoque un branchement vers un code de dérivation associé. Ce premier procédé peut également être amélioré en attachant différents niveaux de sécurité aux instructions de dérivation et en leur associant des codes de dérivation d'autant plus complexes (ou défensifs vis à vis d'attaques de sécurité décrites ci-dessus) que leur niveau de sécurité est élevé.

Concernant la première variante, le deuxième mode d'introduction de "codes de dérivation" consiste à introduire le code de dérivation dans la mise en oeuvre de l'interpréteur lui-même: entre l'exécution de deux instructions consécutives du code, l'interpréteur exécute le code de dérivation, soit de manière systématique, soit de manière sélective, soit de manière aléatoire. Il peut par exemple exécuter ce code seulement lors de l'appel de certaines méthodes (typiquement de bibliothèques, dites API "Application Program Interface") sensibles.

L'avantage du premier mode est de permettre d'introduire les exécutions de code de dérivation de manière sélective, ce qui conduit à une moindre pénalisation en terme de temps d'exécution si le nombre de telles dérivations est faible. Il permet également la mise en oeuvre de politiques de sécurité dites "discrétionnaires", c'est à dire à la discrétion des applications.

En revanche, le second mode sera plus avantageux si le nombre de dérivations souhaitées est important car la mise en oeuvre du procédé dans l'interpréteur lui-même pourra alors être optimisée. Par ailleurs, il permet la mise en oeuvre de politiques de sécurité dites "mandataires" où les contrôles sont imposés de manière uniforme à toutes les applications.

Les deux précédents susdits modes d'introduction nécessitent l'introduction d'un code de dérivation. L'invention propose quatre modes pour réaliser ces

codes de dérivation de manière à ce qu'ils introduisent des variantes dans les temps d'exécution et les empreintes physiques mesurables.

Concernant la première variante, le premier mode de réalisation de "codes de dérivation" à empreinte physique et durée variable consiste à effectuer un calcul dit "superflu" dépendant de données connues à l'exécution (qui peuvent donc différer à chaque exécution). Ce calcul superflu doit être sans effet sur le résultat final de l'exécution de l'interpréteur. Un exemple simple de tel calcul est un test de parité d'une donnée dynamique (connue à l'exécution) qui peut conduire soit à une action vide, soit à l'ajout d'un élément d'une pile suivi de son retrait immédiat. Il est à noter que le nombre d'actions possibles n'est pas forcément limité à deux. Un nombre d'actions possible important conduira à une variabilité importante dans le temps d'exécution et l'empreinte physique du code de dérivation.

Le deuxième mode de réalisation de "codes de dérivation" améliore le premier mode en le dotant d'un tirage aléatoire d'une donnée supplémentaire lors de l'exécution du calcul superflu, ladite donnée supplémentaire étant utilisée dans le calcul effectué par le code de dérivation (par exemple dans un test dudit code). Ce tirage aléatoire ajoute un nouvel élément variable et rend encore moins prévisible le temps d'exécution et l'empreinte physique du code de dérivation.

Le troisième mode de réalisation de "codes de dérivation" améliore l'efficacité des deux précédents en remplaçant le test permettant de décider de l'action suivante par un branchement dans un tableau dit d'indirection, c'est-à-dire contenant les adresses des actions possibles, à un indice calculé à partir des éléments variables (donnée dynamique et/ou résultat d'un tirage aléatoire).

Le quatrième mode de réalisation de "codes de dérivation" améliore le premier mode (et par conséquent les trois autres) en considérant un calcul superflu qui,

tout en restant sans effet sur le résultat final, présente les caractéristiques externes (empreinte physique) d'un calcul sensible particulier (par exemple chiffrage ou déchiffrage) sans rapport avec le code effectif de l'application. Un tel calcul superflu permet de tromper un attaquant qui tenterait de déduire des secrets en mesurant l'effet physique de l'exécution de l'application. Un tel procédé peut être qualifié de "leurre logiciel" puisqu'il a pour objet d'induire en erreur les attaquants en leur faisant croire à la présence dudit calcul sensible dans le code effectif de l'application. Ce mode peut être réalisé simplement en mettant en œuvre le calcul sensible en question sans retenir son résultat.

Concernant la deuxième variante, le premier mode d'introduction de "mises en œuvre multiples" de certaines instructions consiste à enrichir l'ensemble des instructions reconnues par l'interpréteur avec une pluralité de mises en oeuvre pour une instruction donnée. Ces mises en oeuvre seront réalisées de façon à avoir des empreintes physiques et des temps d'exécution différents tout en produisant un résultat identique. N'importe laquelle de ces mises en oeuvre peut être utilisée indifféremment dans le code. Cette utilisation peut être effectuée soit manuellement, par programmation, soit automatiquement lors de la génération de code. Dans ce dernier cas, le générateur de code peut être guidé, pour produire ces instructions, par des annotations insérées par le programmeur dans le code source et permettant de désigner des portions de code sensibles (par exemple, et de manière non limitative, des procédures de chiffrement ou de vérification de droits d'accès). Ce premier mode peut également être amélioré en attachant différents niveaux de sécurité aux mises en oeuvre d'instructions et en leur associant des mises en oeuvre d'autant plus complexes (ou défensives vis à vis d'attaques de sécurité) que leur niveau de sécurité est élevé.

Concernant la deuxième variante, le deuxième mode d'introduction de "mises en œuvre multiples" de certaines instructions consiste à inclure dans la mise en

œuvre elle-même de l'instruction un branchement à une portion de code alternatif qui déterminera dynamiquement la mise en oeuvre à exécuter.

L'avantage du premier mode est de minimiser le surcoût en terme de temps d'exécution puisque le choix de la mise en oeuvre d'instruction à appliquer est déterminé préalablement à l'exécution. Il permet également la mise en oeuvre de politiques de sécurité dites "discrétionnaires", c'est à dire à la discrétion des applications.

L'avantage du second mode est de compliquer encore les attaques nécessitant une synchronisation avec le code puisque deux exécutions consécutives de la même instruction (au même emplacement dans le code) seront susceptibles de prendre des temps d'exécution différents et d'offrir des empreintes physiques différentes. Par ailleurs, ce second mode permet la mise en oeuvre de politiques de sécurité dites "mandataires" où les contrôles sont imposés de manière uniforme à toutes les applications.

Les deux modes ne s'excluent pas mutuellement : une réalisation peut inclure une multiplicité de mises en oeuvre pour une instruction donnée, certaines d'entre elles (ou toutes) étant mises en oeuvre par un branchement à une portion de code alternatif déterminant dynamiquement la mise en oeuvre à exécuter.

Le susdit deuxième mode de la deuxième variante nécessite l'introduction d'un code alternatif associé à une instruction. L'invention propose trois modes pour réaliser ce code alternatif de manière à ce qu'il introduise des mises en œuvre différentes dans les temps d'exécution et l'empreinte physique mesurée.

Concernant la deuxième variante, le premier mode de réalisation de "codes alternatifs" à empreinte physique et durée variable consiste à proposer une pluralité de mises en oeuvre différentes de l'instruction et de conditionner le

choix de la version exécutée à un test dynamique, c'est à dire dépendant de données connues à l'exécution. Un exemple simple de tel calcul est un test de parité d'une donnée dynamique (connue à l'exécution). Un nombre de mises en oeuvre important conduira à une variabilité importante dans le temps d'exécution et dans l'empreinte physique du code alternatif.

Le deuxième mode de réalisation de "codes alternatifs" améliore le premier mode en le dotant d'un tirage aléatoire d'une donnée qui est ensuite utilisée pour la réalisation du test conduisant au choix dynamique de la version exécutée. Ce tirage aléatoire ajoute un nouvel élément variable et rend encore moins prévisibles le temps d'exécution et l'empreinte physique du code alternatif.

Le troisième mode de réalisation de "codes alternatifs" améliore l'efficacité des deux précédents en remplaçant le test permettant de décider de la version choisie par un branchement dans un tableau d'indirection (contenant les adresses des versions disponibles) à un indice calculé à partir des éléments variables (donnée dynamique et/ou résultat d'un tirage aléatoire).

Ainsi, l'introduction de variantes dans les temps d'exécution des codes interprétés et donc les empreintes physiques permet de rendre plus difficiles les attaques reposant sur les dites empreintes physiques, en faisant en sorte qu'une action codée dans l'implémentation de l'application puisse avoir des signatures électroniques différentes et se produisant à des temps d'exécution variables.

L'implémentation des susdits codes interprétés sera effectuée sur des modules d'interprétation de code logiciel comme des machines virtuelles de la famille JAVA, et sur des modules d'interprétation de code physique du type micro-contrôleur ou micro-processeur.

REVENDICATIONS

1. Procédé de sécurisation des systèmes informatiques comprenant au moins un module d'interprétation de code et des capacités mémoire de stockage du code interprété présentant des empreintes physiques mesurables, caractérisé en ce que dans le but de rendre plus difficiles les attaques reposant sur des mesures physiques ou nécessitant une synchronisation avec le susdit code interprété, il consiste à introduire des variantes d'exécution du code interprété, lesdites variantes ayant un effet sur les temps d'exécution du code interprété ou ses empreintes physiques mesurables.

2. Procédé selon la revendication 1,
caractérisé en ce qu'il comprend des dérivations vers de nouvelles portions de code, dits "codes de dérivation", qui n'appartiennent pas au code d'origine.

3. Procédé selon la revendication 1,
caractérisé en ce qu'il comprend une pluralité de mises en œuvre de certaines instructions, chacune réclamant un temps d'exécution différent ou présentant une empreinte physique différente tout en fournissant un résultat identique.

4. Procédé selon la revendication 2,
caractérisé en ce qu'il comprend un premier mode d'introduction de "codes de dérivation" consistant à introduire une (ou des) instruction(s) spécifique(s) à certains endroits particuliers du code, soit manuellement soit automatiquement lors de la génération du susdit code.

5. Procédé selon la revendication 4,
caractérisé en ce que les instructions de dérivation sont associées à des niveaux de sécurité qui correspondent à des degrés de complexité de leur code de dérivation, les plus complexes étant considérés comme les plus défensifs

vis à vis d'attaques de sécurité nécessitant une synchronisation avec le code ou une mesure de son empreinte physique.

6. Procédé selon la revendication 2,

caractérisé en ce qu'il comprend un deuxième mode d'introduction de "codes de dérivation" consistant à introduire le code de dérivation dans la mise en oeuvre de l'interpréteur lui-même.

7. Procédé selon la revendication 6,

caractérisé en ce que le code de dérivation introduit dans la mise en œuvre de l'interpréteur est exécuté soit de manière systématique par l'interpréteur, soit de manière sélective, soit de manière aléatoire.

8. Procédé selon la revendication 2,

caractérisé en ce qu'il comprend un premier mode de réalisation de "codes de dérivation" consistant à effectuer un calcul dit "superflu" dépendant de données connues à l'exécution.

9. Procédé selon la revendication 2,

caractérisé en ce qu'il comprend un deuxième mode de réalisation de "codes de dérivation" consistant à doter le susdit premier mode d'un tirage aléatoire d'une donnée supplémentaire lors de l'exécution du calcul superflu, ladite donnée supplémentaire étant utilisée dans le calcul effectué par le code de dérivation.

10. Procédé selon la revendication 8,

caractérisé en ce que le susdit premier mode de réalisation de "codes de dérivation" est amélioré en attachant différents niveaux de sécurité aux mises en œuvre d'instructions et en leur associant des mises en œuvre d'autant plus complexes.

11. Procédé selon la revendication 2,
caractérisé en ce qu'il comprend un troisième mode de réalisation de "codes de dérivation" consistant à remplacer dans les susdits premier et deuxième mode le test permettant de décider de l'action suivante par un branchement dans un tableau d'indirection contenant les adresses des actions possibles à un indice calculé à partir des éléments variables (donnée dynamique et/ou résultat d'un tirage aléatoire).

12. Procédé selon la revendication 2,
caractérisé en ce qu'il comprend un quatrième mode de réalisation de "codes de dérivation" consistant à effectuer un calcul superflu présentant les caractéristiques externes d'un calcul sensible particulier.

13. Procédé selon la revendication 3,
caractérisé en ce qu'il comprend un premier mode d'introduction d'une pluralité de mises en oeuvre de certaines instructions consistant à enrichir l'ensemble des instructions reconnues par l'interpréteur avec une pluralité de mises en œuvre pour une instruction donnée ; les susdites instructions sont effectuées soit manuellement, par programmation, soit automatiquement lors de la génération du code.

14. Procédé selon la revendication 3,
caractérisé en ce qu'il comprend un deuxième mode d'introduction de la susdite pluralité de mises en oeuvre de certaines instructions consistant à inclure dans la mise en œuvre elle-même de l'instruction un branchement à une portion d'au moins un code alternatif à empreinte physique ou durée variable qui détermine dynamiquement la mise en oeuvre à exécuter.

15. Procédé selon la revendication 14,
caractérisé en ce qu'il comprend un premier mode de réalisation du susdit code alternatif consistant à proposer une pluralité de mises en oeuvre

différentes de l'instruction et en conditionnant le choix de la version exécutée à un test dynamique, c'est à dire dépendant de données connues à l'exécution.

16. Procédé selon la revendication 14,
caractérisé en ce qu'il comprend un deuxième mode de réalisation du susdit code alternatif consistant à améliorer le susdit premier mode de réalisation de "codes alternatifs" en le dotant d'un tirage aléatoire pour la réalisation du test conduisant au choix dynamique de la version exécutée.

17. Procédé selon la revendication 14,
caractérisé en ce qu'il comprend un troisième mode de réalisation du susdit code alternatif consistant à améliorer les susdits premier et deuxième modes de réalisation de "codes alternatifs" consistant à remplacer le test permettant de décider de la version choisie par un branchement dans un tableau d'indirection contenant les adresses des versions disponibles à un indice calculé à partir des éléments variables.

18. Procédé selon la revendication 1,
caractérisé en ce qu'il est implémenté sur un module d'interprétation de code logiciel dit machine virtuelle.

19. Procédé selon la revendication 18,
caractérisé en ce que ladite machine virtuelle est une plateforme Java.

20. Procédé selon la revendication 1,
caractérisé en ce qu'il est implémenté sur un module d'interprétation de code physique.

21. Procédé selon la revendication 1,
caractérisé en ce qu'il est implémenté sur un système embarqué et sur un module d'interprétation du type micro-contrôleur ou micro-processeur.